Peter Anteater panteate Seat: TBD

CompSci 260P Exam 1 Diagnostic Exam 2 Spring 2023

DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO

SILENCE AND STORE ALL ELECTRONICS

This is a diagnostic exam intended to help you evaluate your readiness for the real exam.

The following rules apply to you, whether you think they do or not. Read and understand them; failure to abide by these rules, or directions given by course staff during the exam, may result in disciplinary action, including but not limited to a failing grade in the class.

- This exam is solely for students enrolled in this lecture. Anyone not enrolled in this lecture may not take an exam.
- You may not open the exam or begin writing until the instructor has explicitly given you permission to do so.
- Keep your UCI ID readily accessible during the test. Proctors may request to see it.
- This exam is closed book, closed notes, and is individual effort. Once course staff begin passing out exams, you may not communicate with anyone other than proctors for any reason, nor may you have electronics, including calculators watches and phones, available to you during the test for any reason. **YOU DO NOT NEED A CALCULATOR!**
- If you leave your seat during the test for any reason, your instructor may collect it and deem you to have turned it in. Do not ask proctors for an exemption to this, they are not authorized to grant such.
- If you are still seated at 7:35, you may not leave your seat until explicitly dismissed by the instructor. Leaving after 7:35 and before being dismissed may result in a grade penalty.
- If you believe a question is ambiguous, write at least two reasonable interpretations and indicate clearly which one you will be using. Then answer your question with that assumption. Unless your interpretation makes the problem much more trivial than intended, we will grade your response as if one of us had made that clarification.
- The purpose of the real exam is to evaluate how well *you* understand the material presented in the course. It is an academic integrity violation to do anything that subverts the goals of this assessment including, but not limited to, not doing your own work or submitting that of anyone else.
- We will only grade responses marked in the space provided for each question.

Nothing you write on this page will be graded. The next page in this booklet contains a spot to answer these questions. You may use this page as scratch paper if you would like, and room to do so exists.

I was investigating the possibility of giving out candy at office hours and ran into the following problem. Suppose I wanted to give each student one piece of candy for each office hours of mine that they attend. I have a candy jar in my office that can hold C pieces of candy. At any point, I can ask the department to add candy to the jar; they'll charge me a fixed price $\$P$ for doing so, regardless of how many pieces of candy I ask them for. They also don't like letting candy stay in the jar, getting stale; to discourage me from letting that happen, they charge me $\$d$ for each leftover piece after every office hour.

For the next quarter, I have n office hours planned; at the ith office hour, I project that s_i students will show up, each of whom will get one piece of candy. For this problem, assume that the projected attendance at office hours is accurate. At the start of the quarter, my candy jar is empty, and I aim to have it empty at the end of the semester. My goal is to provide one piece of candy to each student that shows up at each office hour, while spending the least amount of money and having an empty candy jar at the end of the quarter. Give a dynamic programming algorithm to determine the minimum cost for me to do so.

While it is possible to solve this problem in time polynomial only in n, full credit will be given for a correct solution with running time $\mathcal{O}(nC)$.

Remember to write your answer on the answer page →, not here, although you may use this space for scratch work.

To complete these instructions, do the following.

- Give a clear and precise English definition that describes the function you are implementing. Not *how* it works (yet), but rather *what* it solves.
- Give that function a meaningful variable name. This is not [just] me being pedantic; I have found it helps students with this topic if they do this. "OPT" is not a meaningful variable name, nor is "table." Single letters are not meaningful variable names.
- Give a clear recursive formula or algorithm in terms of smaller instances of *exactly* the same problem.
- Describe the iterative running time correctly. This can either be by writing the iterative algorithm (in which case, you can point out where the previous part is within the solution), or by taking your recursive solution, counting the cases, describing the order in which the table would be filled in, and analyzing the time accordingly.

Write your answer to question 1 on this page

Nothing you write on this page will be graded. The next page in this booklet contains a spot to answer these questions. You may use this page as scratch paper if you would like, and room to do so exists.

Read the following prompt. There are two problems described. One of them can be solved in polynomial time. The other is known to be $\mathcal{NP}$-complete .

You will need to identify which one is which, indicate why one can be solved in polynomial time, and indicate why the other one is likely to be computationally difficult via reduction. For the last part, you may assume any problem shown in class (lecture or lab) to be $\mathcal{NP}$-complete is. The following are a few such problems:

- Independent Set: Given a graph G and an integer k, determine if there is some subset of the vertices V', $|V'| \geq k$ such that no two vertices in V' share an edge.
- 3-Color: Given an undirected graph G, determine whether there is a mapping of vertices in G to three distinct colors such that no edge is monochromatic.
- Hamiltonian Path: does this graph contain a simple path that includes every vertex? This is $\mathcal{NP}$-complete for both directed and undirected graphs. It is also $\mathcal{NP}$-complete to check for Hamiltonian Cycles (cycles that include every vertex) in either directed or undirected graphs.
- Subset Sum: given a list of numbers S and a target number T, is there a subset $S' \subseteq S$ whose elements add to T?

Your reduction may not be based on a problem not shown to be $\mathcal{NP}$-complete in lecture or lab.

.....

For each of the following problems, we are given an undirected, weighted graph $G = (V, E)$, a positive integer k, and a positive integer T. In both problems, we need to partition the vertices into k non-empty sets. You may assume all weights are positive if you would like to do so.

- The *spacing* of a clustering is the minimum, over all pairs of vertices that are in *different* sets, of the distance between them. The question of the Maximum Spacing Clustering problem is whether or not a partitioning of the vertices exists with spacing *at least* T. If this were an optimization problem, we would want to find a clustering with the largest possible spacing.
- The *diameter* of a clustering is the maximum, over all pairs of vertices *that are in the same set*, of the shortest path distance between them. The question of the Low-Diamater Clustering problem is whether or not a partitioning of the vertices exists with diameter *at most* T. If this were an optimization problem, we would want to find a clustering with the smallest possible diameter.

Remember to write your answer on the answer page →, not here, although you may use this space for scratch work.

Which version is polynomial-time solvable? Circle exactly one :

Maximum Spacing Clustering Low Diameter Clustering

Briefly justify your answer:

. . . . . .

Which version is $\mathcal{NP}$-complete ? Circle exactly one :

Maximum Spacing Clustering Low Diameter Clustering

Provide a reduction.